COMPONENT BASED FLIGHT SIMULATION IN DIS SYSTEMS

Krzysztof Mieloszyk, Bogdan Wiszniewski
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
krzymi@due.mech.pg.gda.pl, bowisz@eti.pg.gda.pl

Abstract Distributed interactive simulation constitutes an interesting class of information systems, which combine several areas of computer science enabling each individual simulation object to visualize dynamic states of all distributed objects participating in the simulation. Objects are unpredictable and must exchange state information in order to correctly visualize a dynamic 3D scene from their local perspectives. In the paper, a component based approach developed in the ongoing project at GUT[1], has been described; it can reduce the volume of state information being exchanged without losing messages essential for reliable execution of simulation scenarios.

Keywords: Distributed objects, remote state monitoring

Introduction

Distributed Interactive Simulation (DIS) systems form an important application class of collaborative computing environments, in which many independent and autonomous simulation objects, real objects and human operators are connected to one computational framework. In may be for example a local cluster of helicopter flight simulators in a lab with a group of real tanks operating in a remote shooting range, a city traffic simulation system where cars and traffic lights are simulated but some intersections are operated by real policemen, or a complex building on a simulated fire seen on computer screens at the command center, and real firemen on a drill. Any such system performs specific computations, which are unpredictable, have no algorithmic representation, and because of participating real objects, all events must be handled in real-time despite of the system geographical distribution.

Objects participating in a simulation exercise are sending updates on their local state to other objects in irregular intervals. If the updates were sent just in periodic samples, a network supporting any realistic DIS system with many objects would soon be overloaded. Moreover, increasing dynamics of reporting

objects would imply higher sampling rate and would make the performance problems even worse. Delayed (or lost) messages would certainly make any visualization unrealistic. However, if a simulated object dynamics could be estimated with some function of time, the number of messages to be sent would be limited, since "new" states would be calculated by a receiving object instead of sending them out by the reporting object.

This paper reports on the project started at TUG in 2002 and aimed at developing a DIS system with time-critical constraints, involving simulated flying objects (helicopters) and ground vehicles (tanks) in a 3D space.

1. DIS system architecture

Any DIS system consists of simulators (called *simulation objects*), each one designed to model a specific human operated device or vehicle. Any particular simulator may be operating in a distinct geographical location, and its underlying operating system, software and hardware are usually incompatible to other simulators, preventing direct interaction between them. In order to create a collaborative computing environment a system architecture must enable integration of such objects (called *active participants),* and also provide access for observers (called *passive participants)* with logging and monitoring capabilities. Active participants exchange information to update one another on their states as soon as they change. State updates sent by reporting objects are needed by receiving objects to model a 3D global dynamic virtual scene from their local perspectives. Passive observers usually limit their actions to on-line state tracing and logging for future replay, evaluation of active participants progress in a particular training scenario, as well as collecting data for new training scenarios. A generic architecture of a DIS system is outlined in Figure 1; it involves *communication, service,* and *interaction* layers, with distinct functionality and interfaces, marked with vertical arrows described further on.

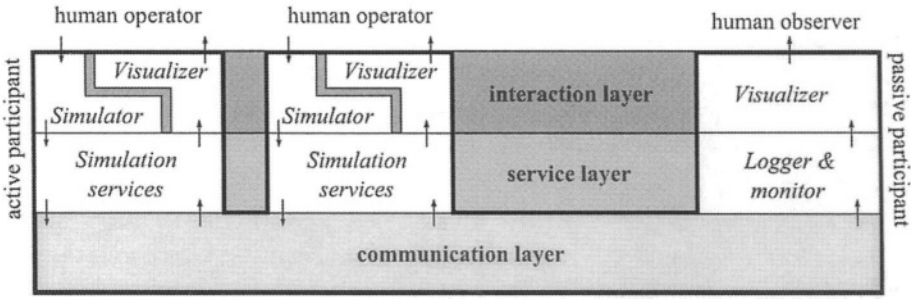

Figure 1. Distributed interactive system architecture

Interaction layer. Human operator provides an external stimulus affecting the internal state of a simulator. According to the semantics of the latter and its current state a new state is determined, reported to the lower layer simulation services, and broadcasted via the communication layer. State updates are received at irregular intervals by simulation services of an interested participant and passed to the visualizer component, which generates (modifies) its local perspective of a global dynamic scene. Based on the view of moving objects outside the cabin and a local state indicated by flight instruments inside the cabin, a decision is made by the human operator (pilot) on the next stimulus (maneuver).

Service layer. Simulation services provided by the service layer enable reduction of the volume of state update messages being sent over the system by active participants. If the simulation object movement (state trajectory) can be described with kinesthetic parameters like acceleration, speed, position, mass, force, moment, etc., state prediction can be based on Newtonian rules of dynamics, using a technique known as *dead reckoning* [Lee2000]. States that can be calculated based on the current reported state do not have to be sent out, as the receiving participant can calculate them anyway. Further reduction of the volume of state updates can be achieved by *relevance filtering* of messages that are redundant with regard to some specific context of the scene, e.g. a reporting object is far away and its movement will result in a pixel-size change at a remote display.

Communication layer. The main job of the bottom layer shown in Figure 1 is to make the underlying network transparent to upper layers. Objects may want to join and leave the simulation at any time, require reliable sending of messages, and need time synchronization. This layer has no knowledge on the semantics of data being sent by simulation objects but has knowledge about the location of participants over the network. Two models of communication have been implemented in the project reported in this paper: one with a *dedicated server* and another with *multicast* [MKKK2003]. The former (server based) enables lossless communication and make data filtering easier, but the cost is that each message has to go through the server and a network load increases when many participants work in the same local area network. The latter is scalable, but requires implementation of dedicated transmission protocols on top of the existing unreliable UDP protocol.

2. Component interaction model

Since simulation objects have to invoke specialized services of the communication layer, rather then to communicate directly with each other. The communication layer must implement a standard, system-wide functionality.

For example, *High Level Architecture* (HLA) standard [HLA] requires delivery of such services as: *federation management* for creating, connecting, disconnecting and destroying simulation objects in the system, *time management* for controlling logical time advance and time-stamping of messages, and *declaration management* for data posting and subscribing by simulation objects.

Reduction of the volume of data being sent by objects is achieved by a *dead reckoning* technique, which basically extrapolates new position of an object using its previous position and state parameters such as velocity or acceleration. If object movements are not too complex, the amount of messages to be sent can be significantly reduced [Lee2000] However, the method developed in the reported project utilizes a notion of a semantics driven approach to message filtering, based on maneuver detection, allowing for further reduction of the space of states to be reported. This has been made possible by introducing operational limits characterizing real (physical) objects (vehicles) [OW2003]. We will refer to this method briefly when presenting below another important concept introduced in the reported project, which is *component based simulation.*

In order to build and run a simulation system, the reported project required simulators of various physical objects of interest. They had to be realistic in the sense of their physical models, but allowing for easy configuration and scalability of simulated vehicles. This has been achieved by adopting the concept of a physical component shown in Figure 2a.

A component has its local state $\vec{S}$, set initially to some predefined value. Upon the external stimulus $\vec{x}$ coming from the operator or other component its new (resultant) state $\vec{S}'$ is calculated as $\vec{S}' = G(\vec{S}, \vec{x})$, where G represents a state trajectory of the simulated component, given explicitly by a state function or implicitly by a state equation. Subvector $F(\vec{S})$ of the resultant state is reported outside to other components (locally) or other simulators (externally), where F is a filtering function, selecting state vector elements relevant to other components or simulators.

With such a generic representation a component may range from the body with a mass, airfoil objects, like a wing, rotor or propeller, through various types of engines generating power and momentum, up to undercarriage interacting with a ground.

Simulation object

The main idea behind the component based approach is to divide a simulated object into most significant units, and then to simulate each one separately. This approach allows for *flexibility,* since simulators can be readily reconfigured by changing parameters of a particular component (or by replac-

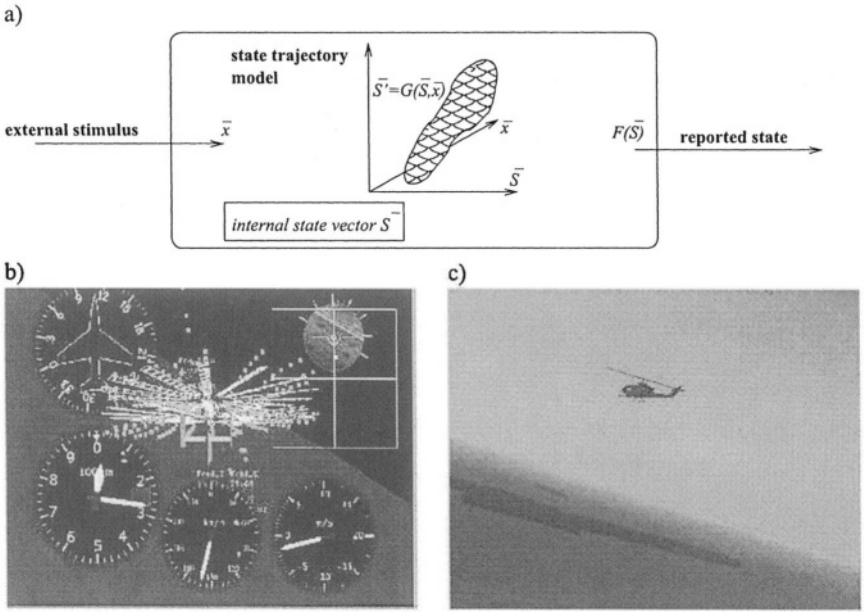

Figure 2. Simulation object: (a) generic component (b) view of components (c) remote view

ing one with another), as well as *parallelization,* since components may run on clusters if more detailed calculations are required.

With the external stimulus, the user can influence behavior of the component work by changing some of its parameters. Reported state vector $F(\vec{S})$ can affect the state of other components of the simulation object. Based on all states, control parameters and the semantics of a component, it is possible to calculate the external state vector as an influence of the component on simulated object. After combining all state vectors reported by components of the simulation object, it is possible to define its resultant behavior.

Consider for example two cooperating components, an engine and a propeller. In order to simulate correctly each component, local state vectors $\vec{S}$ of an engine and a propeller have over 10 elements, and over 15 elements respectively. However, interaction between them can be modeled with just a 2-element vector consisting of a rotation velocity and torque. Similarly, a two element state vector is sufficient to represent cooperation between a helicopter rotor and an engine, despite that simulation of a rotor requires over 25-element state vectors.

The general modeling technique is to describe a simulation object with a graph, of which each single node corresponds to the respective component. For each pair of nodes which can affect one another an arc is drawn and the corresponding reported state vector $F(\vec{S})$ associated. The size of reported state

vectors attributed to individual arcs determine the real volume of data that have to be exchanged between components during simulation. For simulation objects considered in the project, namely ground vehicles, single and twin rotor helicopters, propeller and jet planes, and which may consist of the components described below the size of reported state vectors $F(\vec{S})$ never exceeded two. A sample view of cooperating components of a simulated single rotor helicopter is shown in Figure 2b

Wing. A wing has parameters which describe its dimension, fixing point to the fuselage and the airfoil sections with characteristics combining lift and drag with angle of attack. State of the wing can be affected by the arrangement of ailerons and flaps, and its possible rotation along the longitudinal axis. In this way it is possible to model both lifting and control surfaces of the wing. Additionally, by taking into consideration linear speed of the air stream or angular speed of the wing, the resultant moment and force applied to the simulated object can also be calculated.

Rotor. A helicopter rotor is the most complex component in the project, as it is modeled with several rotating wings (blades). Its state vector elements include dimension of blades, their number, elasticity, induced speed of air flow, airfoil section characteristics, blade fluctuations and angular speed. By changing parameters affecting the collective pitch and periodic pitch, the user (pilot) can control the rotor in the same way as in a real helicopter [SN2002]. Reported state vector $F(\vec{S})$ consists of the resultant forces and moments calculated at over a dozen points evenly distributed over a rotor disk. It is also necessary to consider torque, which is required to determine correctly the state of the entire power transmission system.

Propeller. This component is a simplified version of a helicopter rotor, based on the same semantics and parameters. Elasticity and fluctuations of blades are neglected in calculating of $F(\vec{S})$, but the parameter describing the collective pitch setting is added. The internal state vector of a propeller is the same as in the rotor component.

Engine. This component supports the semantics of both, a jet turbine and a piston engine. Including the internal state vector describing angular speed, working temperature and the maximum power, the user can control its behavior by setting up a throttle. Calculation of the reported state vector $F(\vec{S})$ requires gathering torque values of all attached components, like a propeller or rotor, to calculate the resulting angular speed for the entire power transmission unit taking into account its inertia.

Undercarriage. It is the only component that allows the simulated object to interact with a ground. The internal state vector describes the radius of a tire, shock-absorber lead, and its elasticity, as well as speed of the entire plane, and the relative location of the undercarriage with regard to the plane (helicopter) body. This component has its semantics, defined by a characteristic describing interaction patterns between the tire and the absorber during the contact with a ground. By changing the angle of turn of the wheel and the braking factor, it is possible to control the traction over the runway. As with other components, the reported state vector $F(\vec{S})$ describes the moment and the reaction force of the ground applied through the undercarriage to the simulated object body.

Remote object interaction

As mentioned before any simulation object in a DIS system sends out updates on its state changes to enable other (remote) objects to calculate its position in global scene from the local perspective of each one. The volume of messages is reduced by adopting a dead reckoning scheme, allowing calculation of some "future" states based on current states. While dead reckoning applies mostly to calculating trajectories of moving objects, further reduction of the volume of information being sent is possible based on specific relationships between various elements of the material object state vector. A sample view of a remote object's state (a helicopter) from the local perspective of another object (also a helicopter) is shown in Figure 2c

Active participants. State vector $F(\vec{S})$ reported by each component locally may allow a certain degree of redundancy, depending on the specific internal details of the simulation object. However, the reported state (update) sent out to remote objects must use a state vector in a standard form. In the current implementation it consists of position $\vec{L}$, orientation $\vec{O}$, linear velocity $\vec{v}$, linear acceleration $\vec{a}$, angular velocity $\vec{\omega}$, angular acceleration $\vec{\gamma}$, resultant force $\vec{F}$, and resultant moment $\vec{M}$. In a properly initiated simulation system, where each receiver (observer) has once got full information about each participant, for objects associated with decision events (maneuvers initiated by their human operators) only changes in their acceleration are needed [OW2003].

Passive participants. State prediction is less critical for passive participants, as they do not interact (in the sense of providing a stimulus) with other objects. They do not have any physical interpretation and there is no need to inform users about their existence in a system. They may be independent 3D observers, like a hot-air balloon, or a 2D radar screen, or a map with points moving on it. Their only functionality is monitoring and/or logging the traffic of state update messages. In a DIS system implemented in the project a logger has been introduced. Based on the recorded log entries it can create simulation

scenarios, which can be next edited, modified and replayed in the system. In that particular case the logger may temporarily become an active participant.

Human operator

In order to implement any realistic DIS scenario involving "material" objects two problems must be solved. One is state predictability, important from the point of view of the volume of state update messages, and another is object ability to perform maneuvers within specific limits imposed by its operational characteristics. Each object having a mass and characterized with kinesthetic parameters behaves according to the Newtonian laws of dynamics. Classes of behavior that such a material object may exhibit are described by basic equations combining these parameters (function $G(\vec{S})$. Such a form of object representation, although true from the point of view of physics is far too detailed from the point of view of simulating exercises with real flying objects controlled by humans. It has been argued [OW2003] that by introducing the notion of a *maneuver* and *operational characteristics* of simulation objects, the space of possible states to be considered can be significantly reduced. In consequence, there are less states to predict and the flow of state update messages can be reduced further.

State predictability. The "logic" of flight may be described with a simple automaton involving just five states representing human operator (pilot). The basic state of a flying object is *neutral,* i.e. it remains still or is in a uniform straight line motion. According to the first Newton's law of dynamics both linear and angular accelerations are zero, while the linear velocity is constant. An object in a neutral state may start *entering* a new maneuver and keep doing it as long as its linear or angular acceleration vary. This may eventually lead to a stable state, which is the actual maneuver; in that case both linear and angular acceleration vectors of the object are constant and at least one of them must be non-zero. Any subsequent increase or decrease of any of these acceleration vectors implies further *entering* or *exiting* a maneuver. Exiting a maneuver may end up with entering another maneuver or returning to a neutral state. There is also a *crash* state, when at least one of the object parameters exceeds its allowed limits, e.g. exceeding a structural speed of the airplane ends-up with its disintegration. It found out in the project, practically each state transition of the automaton described above can be detected just by tracing changes of angular or linear acceleration.

Operational characteristics. All components described before has realistic operational limits, usually listed in the user's manual of the simulated object. The mass may vary, but stay between some minimum (empty) and maximum (loaded) values. There are several speeds characterizing a flying object, e.g.

for planes it is the minimum (stall) speed for each possible configuration (flaps up or down, landing gear up or down), maximum maneuvering speed to use in maneuvers or turbulent air, and maximum structural speed not to be exceeded even in a still air. Resultant lift and drag forces for the wing are the function of the airflow speed and angle of attack, which may change up to the critical (stall) angle, specific to a given profile. Finally thrust is a function of engine RPMs, which may change within a strictly defined range of $[min,max]$ values. Based on these parameters, and a maneuver "semantics" described before, it is possible to calculate (predict) most of the in-flight states intended by the human operator, excluding only random and drastic state changes such as mid-air collision or self-inflicted explosion.

3. Summary

In the current experimental DIS application three classes of simulation objects have been implemented using components described in the paper: a tank, a light propeller airplane, and two kinds of helicopters, with single or twin rotors. The notion of a generic component introduced in Figure 2a proved to be very useful. Current development is aimed at expanding the concept of components on vessels, which besides a propeller-like component and engine, require a body model, simple enough to avoid complex computations but precise to describe interactions between the hull and surrounding water.

Notes

1. Funded by the State Committee for Scientific Research (KBN) under grant T-11C-004-22

References

[SN2002] Seddon J. and Newman S. (2002). *Basic Helicopter Aerodynamics* Masterson Book Services Ltd.

[HLA] DoD. *High Level Architecture interface specification.* IEEE P1516.1, Version 1.3. http://hla.dmso.mil.

[Lee2000] Lee B.S., Cai W., Tirner S.J., and Chen L. (2000). Adaptive dead reckoning algorithms for distributed interactive simulation. *I. J. of Simulation,* 1(1-2):21–34.

[MKKK2003] Mieloszyk K., Kozlowski S., Kuklinski R., and Kwoska A. (2003). Architectural design document of a distributed interactive simulation system KBN-DIS (in Polish). Technical Report 17, Faculty of ETI, GUT.

[OW2003] Orlowski T. and Wiszniewski B. (2003). Stepwise development of distributed interactive simulation systems. In *Proc. Int. Conf. Parallel and Applied Mathematics, PPAM03,* LNCS, Springer Verlag, to appear.